Tolerating Partial Failures on IEC 61499 Applications

Mário de Sousa, Christos Chrysoulas, and Aydin Homay
INESC TEC - INESC Technology and Science, and
Electrical and Computer Engineering Dept, Faculty of Engineering, University of Porto
{msousa, chech, homay}@fe.up.pt

ABSTRACT
In a modern industrial environment control programs are distributed among several devices. This raises new challenges, especially in handling the failure modes. Automatic reconfiguration is a possible approach in dealing with partial failures. The authors have proposed an IEC 61499 based replication framework for building fault tolerant applications - thus a failure of one sub-component does not to jeopardize the execution of the whole application. The proposed framework is capable of supporting dynamic reconfiguration on all automation levels in an industrial cell in order to patronage a fault tolerant and seamless production line.

Keywords
IEC 61499, fault tolerance, replication.

1. INTRODUCTION
In today's industrial environments there is an increasing need for building flexible and robust systems. Traditional manufacturing systems were typically used on a firm/ centralized hierarchy of programmable logic devices (PLCs). In order to reconfigure such a system, the plant should be shut-down and the components should be rewired and reprogrammed. A modern and flexible industrial system should be able to quickly adjust to the changes taken place, meaning being able to adapt and reconfigure based on the changing environments. It must also be able to react in real time while still being safe and reliable.

IEC 61499 [1] was introduced as the standard that is going to provide the means for the development of distributed control applications. The main novelty brought by IEC 61499 is an event driven execution approach, which provides the synchronisation primitives between the sub-applications that compose the distributed control application, while its predecessor IEC 61131-3 never really managed to cope with the distributed nature of the modern industrial applications. In a distributed environment, partial failures or break downs during the execution of the program may occur, and therefore the developer must take into consideration how the application will react when these partial failures happen. These new failure modes should be taken into account, especially in safety-critical applications. A typical approach to overcome these failures is based on building fault tolerant applications based on redundancy. By introducing redundancy, the failure of one sub-component of the distributed system will be masked by the continued execution of its redundant partner.

The authors define a framework in which this approach may be applied in the IEC 61499 environment. This framework takes special care in guaranteeing that all replicas are kept with the same synchronized internal state, so that changing from one replica to another does not impact the remainder of the distributed application. Additionally, the framework allows the use of partial replication, where the application developer may choose to only introduce replication in the most critical software components. The proposed framework can easily be applied for re-configuration purposes.

In this paper, an overview of IEC 61499 standard is presented in section 2. The proposed framework is summarized in section 3. An example application is used to explain the proposed framework in section 4. An implementation of the replication framework on the FORTE [2] IEC 61499 execution environment is also presented in section 4, alongside an initial basic test application. A full validation example based on the Baggage Handling System is presented in section 5. Related work can be found in section 6. Conclusions and future work are presented in section 7.

2. IEC 61499 OVERVIEW
The IEC 61499 standard, the successor of IEC 61131-3, is fitted with features that make it more suitable for supporting dynamic reconfigurable control systems. Control applications consist of a network of Function Blocks (FBs) that exchange data and events. The FBs execute algorithms and encapsulate internal state (variables and a state machine). FBs are represented graphically by a rectangle containing the data inputs (left) and outputs (right), to which is added a second header rectangle for the event interface (examples in Figure 4).

The event triggered nature of IEC 61499 makes it possible for an execution sequence to be planned in advance, and also reconfiguration applications to be easily triggered by control application events. Note that events may easily propagate in a network of FBs, as a FB that is executing may generate several output events even before its own execution has completely terminated. The FBDK [3] execution environment, for example, executes these events sequentially.

Being event-driven allows FBs not to depend on any external schedule - a block encapsulates the desired function and will behave the same anywhere. If and when an event occurs, the user can choose to have a FB react to this event and carry out a particular calculation or provide some output. It is important to note that FB block systems are made to be run with real system, not just modelling them.

IEC 61499 distinguishes between (a) basic FBs that contain algorithms and internal state, (b) composite FBs built as a network of basic and/or composite FBs, (c) sub-applications - a network of FBs and sub/applications, and (d) applications - also a network of FBs and sub/applications. Unlike the FBs, sub-applications and applications can be distributed amongst different devices and resources [4-5] in order to describe system behaviour. One IEC 61499 system therefore consists of one or more Applications, Devices and Resources. Access to the process and communication interfaces is done through specialized Service Interface FBs (SIFBs). The standard merely defines the interface for SIFBs, allowing for different protocols to be supported by distinct implementations of that same interface.

3. IEC 61499 BASED REPLICATION

Replication in an industrial automation context is typically achieved at the hardware level. An execution device is replicated, and each replica runs an exact copy of the software of the original. We have proposed a more flexible software based replication architecture, by leveraging the modularity of IEC 61499. In this approach the atomicity of replication is a software component - in other words, each software component may be replicated independently of all other software components, leading to more agile and adaptable systems.

Typically, only the most critical software components will be replicated onto several hardware execution devices, whereas the non-critical components may be executed with a single copy. In the proposed replication framework the software component that may be replicated is mapped onto an IEC 61499 FB. The replica may reside on another execution device, or may even be distributed amongst several execution devices (see Figure. 1).

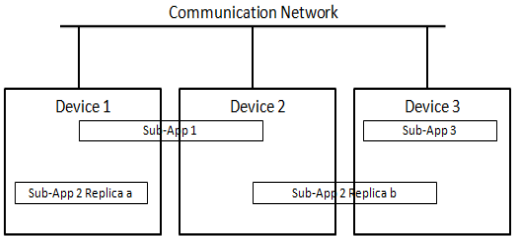

Figure 1. Non-replicated and replicated sub-applications mapping to devices.

Taking into account that not all software components are replicated, several interaction scenarios may be identified. More information can be found in [6].

Voters are widely used with a variety of fault tolerance techniques, including design and data diverse techniques. Voters compare the results of two or more variants, and decide the correct result, if one exists. There are many type of voters [7], and the decision on which voting algorithm to use is dependent on the required application semantics. For the voting to be feasible all replicas must be kept with their internal data structures in synch. Assuming that the algorithms executed by all replicas is deterministic (does not depend, for example, on using a random value), replica synchronisation may be achieved by guaranteeing that all the events and data arrive at all the replicas in the exact same order. To guarantee this ordering, alternative implementations for the communication SIFBs are provided by the replication framework depending on the fault models that need to be tolerated.

Due to the possible varying network delays in the transit of messages over the network, these alternative SIFBs introduce delays in the data that are transmitted faster in order to guarantee that all replicas will receive all data messages in the exact same sequence. This, as explained in the previous paper [6], is known as the timed messages protocol [8]. Fault models that include errors in the communication networks require more costly (slower) communication protocols when compared to communication protocols that may be used when only faults in the execution devices are considered.

4. IMPLEMENTATION - TEST APPLICATION

We implemented this framework on FORTE, an open source IEC 61499 execution environment. Our replication framework has been developed as an extension of the communication SIFBs of FORTE. As can be seen from the previous discussion, the fundamental change is to guarantee that replicas maintain synchronised, thus providing the needed for the smooth operation of the system even when a partial failure occur. FORTE is a program that acts similar to a virtual machine, that only executes IEC 61499 applications. This execution environment is developed together with the 4DIAC development environment, where the programmer may design IEC 61499 applications using a graphical interface. Once programmed, 4DIAC downloads the IEC 61499 applications onto IEC 61499 execution environments using a standardized interface of these execution environments.

All FORTE communication SIFBs are implemented using a layered architecture. We have implemented several new "replication" layers that work between the top layer (typically fbdk) and the bottom layer (usually ip layer). IEC 61499 has several communication FB interfaces and interactions models, including publish/subscribe, and client/server. Each of these can have several implementations, mappings to different communication protocols. Ideally, we should implement a version of each client/server and publish/subscribe that supports replication.

Some simple tests were run based on the trivial XPlus3 sample application that comes with the 4DIAC development environment in order to make an initial validation of the proposed implementation. The original sample application (read input, add 3, print result) simultaneously uses the FORTE and FBDK execution environments. Using the new replication layers a trivial replicated version of this application has been tested. In the replicated version the FB that adds a constant value to the input has been replicated. In principle, both replicas should add the same value, but in our test we are adding distinct values in each replica (3 and 4) so it becomes possible to identify which result was chosen by the voter.

This trivial replicated application was successfully tested with all device instances running on the same computer. Device failures were tested by simply stopping and starting each of the devices running one of the replicas. The results were as expected, where the application was able to produce an output result as long as at least one of the replica devices were executing.

5. FULL VALIDATION SETUP

5.1 Test-bed and System Setup

An example of where the replication framework will be useful is an airport baggage handling system, which is a classic example of a complex distributed automation system that requires high performance, reliability, and flexibility. These systems come with sensors and actuators, embedded control devices and machines, all combined and working in a seamless way. The applicability of using the IEC 61499 reference model in implementing this control architecture has been studied in [9], who created a test bed of more than 50 networked control nodes simulating a small sized airport. The results from the experiments proved that IEC 61499 can meet the needed levels of flexibility.

With a view of increasing the reliability of the proposed solution, we present a fault-tolerant version of this system. Every conveyor in the baggage handling system will be assigned a FB, to be executed by a PLC type device. The most critical conveyors for the seamless function of the whole chain will be controlled by at least two PLC type devices, so that a possible failure in one PLC will not affect the functionality of the chain. For simplicity we are going to use just a small part of a baggage handling system that can be found in airports. Figure 2 gives a view of the actual test-bed to be used for the simulation of a baggage handling system. The proposed simplified system does however contain conveyors for all the major processes in baggage handling, namely check-in,

security scan, identification, and final sorting. A graphical representation can be found in Figure 3.

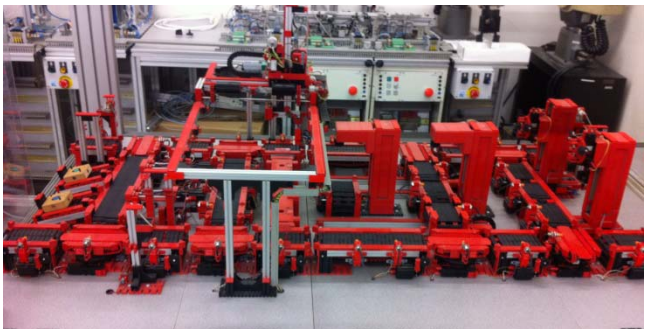

Figure 2. Test-bed for simulating the baggage handling system.

This example contains 4 conveyors. C0 is the bag feeder for C1. There are two possible paths the bags exiting C1 may follow. One is towards C2, and the other one is towards C3. The decision is depending on the feedback the controller of C1 is getting from the Distance Acceptance Stations residing on C2 and C3. The first one declaring that there is no bag in its visual field will be the one qualified to accept a new bag. Figure 4 presents the easiness of connecting and thus mapping FBs to the physical layout.

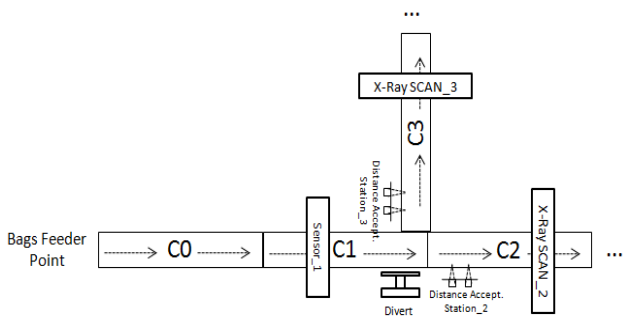

Figure 3. Bag spacing control and diverting system.

So the questions that inevitable come are the following: "What should we replicate in order to avoid system failures? How this replication will help in easily reconfigure our system?"

In this layout, the conveyor that plays the most critical role is C1, since one failure shall disrupt the whole baggage handling process. Thus the replication of functionality of the two aforementioned parts is proposed. The application developer would therefore need to change the original IEC 61499 based control application by simply adding a new instance of the FB controlling this conveyor.

We must now guarantee that each of the replicas is executed in a distinct device, so the failure of one replica does not imply the failure of the other. With this in view, this new replica FB is assigned to a distinct device. Here, the two replicas of C1 are each executed by two different PLCs. These same PLCs also control other conveyors, so the adding of replica FBs to the control application does not necessarily imply an increase in the number of physical devices.

Although IEC 61499 is a framework for distributed applications, the distribution of the application itself is not totally hidden to the application developer, as (s)he needs to insert data communication FBs (namely publishers and subscribers) when distributing an application to run on several execution devices (and resources).

In the case of many to many transmission, the problem of maintaining synchronised internal state is especially evident since we have to introduce replicas for the FB controlling the C1 conveyor, as this FB is receiving data from 2 distinct down-stream conveyors (C2 and C3). In this case we need to guarantee that all messages from the down-stream conveyors reach the C1 replicas in the exact same order. Our replication framework provides an implementation of the Publish and Subscribe SIFBs that enforce the timed messages protocol. So it guarantees the above ordering properties, and the developer implementing the IEC 61499 control application will merely need to change from one implementation of the communication SIFBs to another. The same may be said in the many to 1 communication.

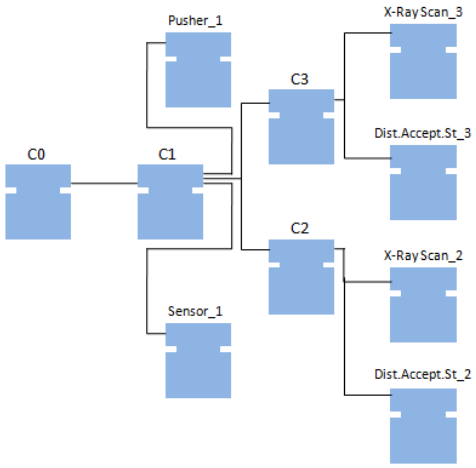

Figure 4. FBs connected following the physical layout.

5.2 Test-bed and System Setup

The FBs controlling the conveyors must send out signals to the conveyor's actuators. However, when using replicas these physical signals are also replicated, and must therefore also be consolidated. In our set-up the physical conveyors include a Remote Terminal Unit (RTU), which basically functions as a physical I/O interface that may be controlled remotely over a network. In our case, we use the Modbus/TCP communication protocol, with the RTUs functioning as Modbus servers.

In our validation application we have all replicas of the same conveyor sending data to that conveyor's RTU. The RTU will function as a voter, placing on the outputs the most recent information received from either replica. By considering that all execution devices follow the fail-stop failure-mode (either work correctly, or stop completely), we can conclude that all working replicas always provide correct outputs, so all information sent to the RTUs will always be correct. If one replica fails, the RTU simply keeps on listening to the single replica that continues sending updated information.

6. RELATED WORK

Little work has been done regarding the use of fault tolerance in the context of IEC 61499 applications. However, somewhat related is the use of IEC 61499 in safety critical applications - [10] applies formalisms to model and validate IEC 61499 applications. Although not in the context of safety-critical application, significant work has been done on formalising the IEC 61499 execution semantics [11-12].

Other related work that focuses on online reconfiguration of IEC 61499 control applications, while guaranteeing the control application's real-time requirements can be found in [13-15].

7. CONCLUSIONS AND FUTURE WORK

This paper is presenting an architecture for building fault tolerant IEC 61499 application based on the concept of the replication of FBs. The authors are arguing that the same methodology can be used for reconfiguration purposes in a distributed industrial environment. The architecture does not require the complete replication of an execution device, but rather supports a more finer grain of control of what parts of the application needs to be replicated. Building industrial systems based on the IEC 61499 framework will provide the needed flexibility to easily perform changes in a production industrial line without the need to shut down the involved industrial cell.

A simple replicated test application is used to initially validate the proposed framework. A full validation setup is in detail presented, demonstrating how the proposed framework can and will be applied to a real world example. Guaranteeing replica determinism between replicas of the same FB is also targeting.

8. ACKNOWLEDGMENTS

This work is financed by the ERDF-European Regional Development Fund through the COMPETE Programme and by National Funds through the FCT-Portuguese Foundation for Science and Technology within project FCT EXPL/EEI-AUT/2538/2013.

9. REFERENCES

[1] International Electrotechnical Commission, “IEC61499-1 ed2.0 Function blocks - Part 1: Architecture”, 2012-11-07.

[2] FORTE Homepage: http://www.fordiac.org/8.0.html (accessed March 13, 2015).

[3] FBDK Homepage: http://www.holobloc.com/doc/fbdk/ (accessed March 13, 2015).

[4] Zoitl, A. 2008.Real-time execution for IEC 61499. Durham, North Carolina, International Society of Automation, 2008.

[5] . Lewis, R.2001. Modelling control systems using IEC 61499. The Institution of Electrical Engineers, 2001.

[6] Sousa, M. 2014.Guaranteeing Replica Determinism on IEC 61499. In 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Barcelona, September 2014.

[7] Pullum, L. L. 2001. Software Fault Tolerance Techniques and Implementation. Artech House computing Library, 2001.

[8] Zhang,S., Burns, A., Chen, J., and . Lee, E.S. 2004. Hard real-time communication with the timed token protocol: Current State and Challenging Problems. Real-Time Systems, Volume 27, Issue 3, pp.271-295, 2004.

[9] Yan,Y., Vyatkin, V. 2011. Distributed execution and cyber-physical design of Baggage Handling automation with IEC 61499. IEEE International Conference on Industrial Informatics (INDIN), 26-29 July 2011, pp.573-578.

[10] Yoong, L. 2010. Modelling and Synthesis of Safety-Critical Software with IEC 61499. PhD Thesis submitted for Electrical and Electronic Engineering, University of Auckland, 2010.

[11] Cengic, G., Ljungkratz, O., Akesson, K. 2006. Formal modeling of Function Block applications running in IEC 61499 execution runtime. In 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Prague, September 2006, pp. 1269-1276.

[12] Dubinin, V., Vyatkin, V. 2008. On Definition of a Formal Model for IEC 61499 Function Blocks. EURASIP Journal of Embedded Systems, 2008.

[13] Sardesai, A.R., Mazharullah, O.,. Vyatkin, V. 2006. Reconfiguration of Mechatronic Systems Enabled by IEC 61499 Function Blocks. In Australasian Conference on Robotcks and Automation (ACRA '06), Aucкland, 6-8 December 2006.

[14] Strasser, T., Müller,I., Sünder C., Hummer, O., Uhrmann, H. 2006. Modeling of Reconfiguration Control Applications based on the IEC 61499 Reference Model for Industrial Process Measurement and Control Systems. In IEEE Workshop on Distributed Intelligent Systems (DIS '06), Prague, June 2006, pp. 127-132.

[15] Zoitl, A., Siinder, C., Terzic, I. 2006. Dynamic Reconfiguration of Distributed Control Applications with Reconfiguration Services based on IEC 61499. In in IEEE Workshop on Distributed Intelligent Systems (DIS '06), Prague, June 2006, pp. 109-114.